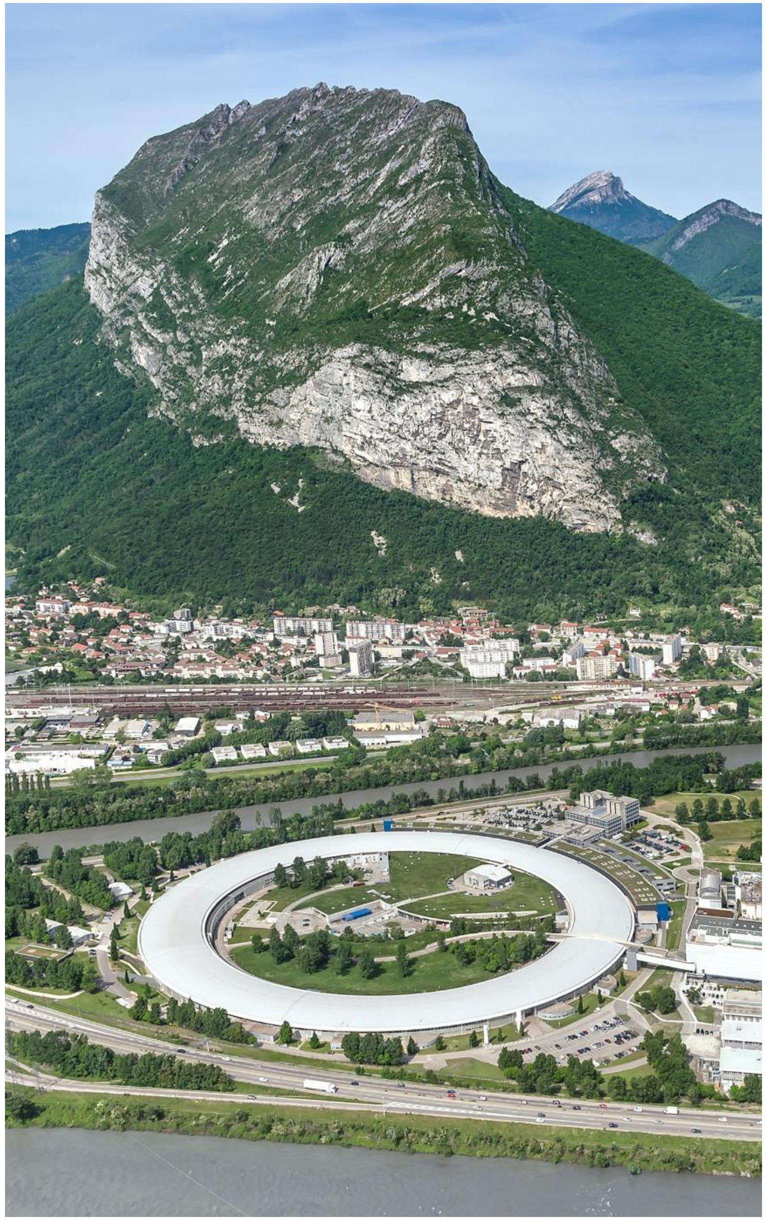


Automatic data processing with EWOKS

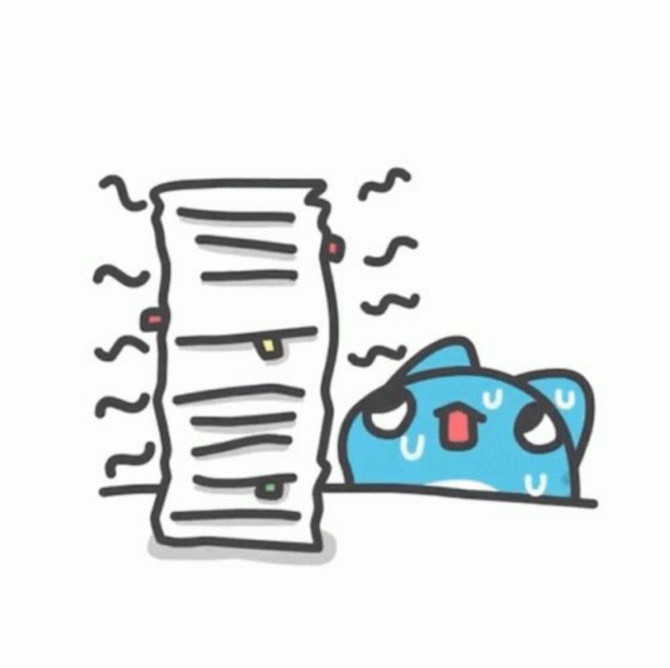
Loïc Huder

ESRF
The European Synchrotron

STREAMLINE has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 870313

WHY AUTOMATIC DATA PROCESSING

With the EBS upgrade and deployment of the BeamLine Instrumentation Support Software (BLISS),
huge amounts of data are acquired



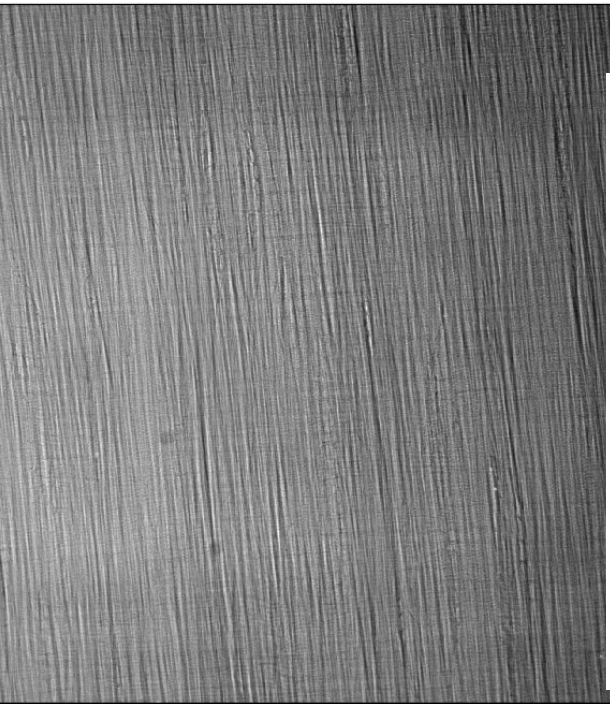
Users and beamline scientists need help
to process this data

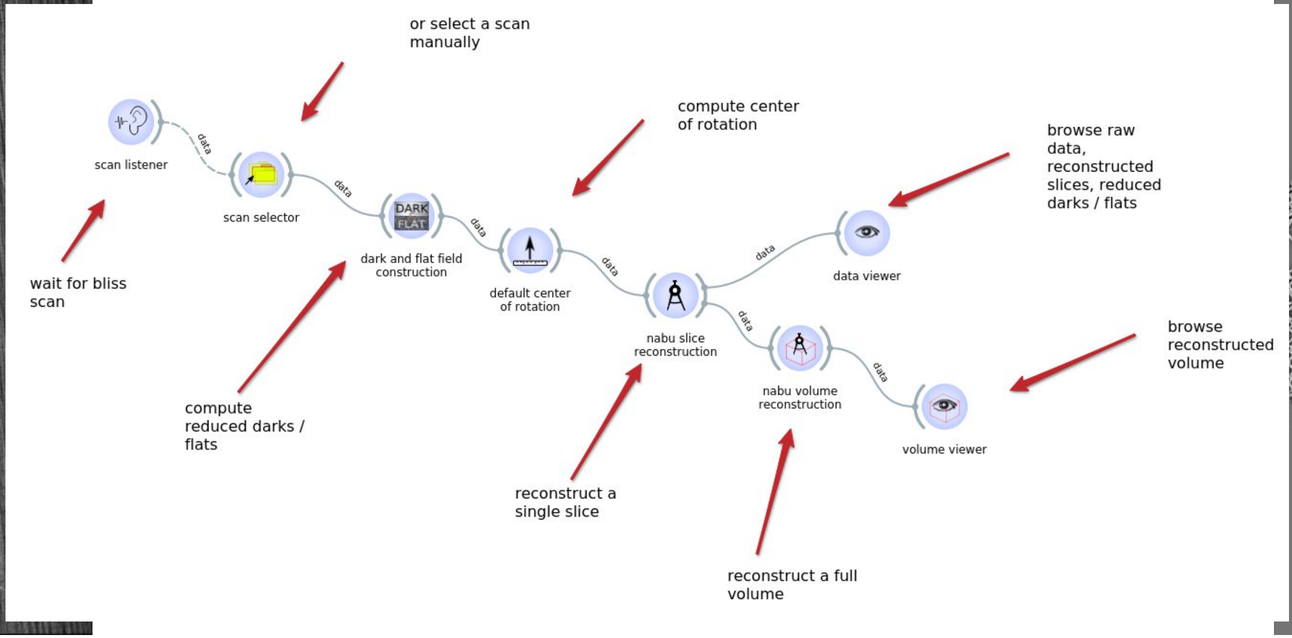
- **Number of publications stays constant**
- **More proposals from non-specialist users**
- **Processed data is the scientific relevant data**

HOW WORKFLOWS CAN HELP

Workflows are data processing pipelines or recipes composed of several steps (tasks)

- Ex: tomography workflow for slice reconstruction




scan listener
data
scan selector
data
DARK
FLAT
dark and flat field construction
data
default center of rotation
data
nabu slice reconstruction
data
data viewer
data
nabu volume reconstruction
data
volume viewer

or select a scan manually

wait for bliss scan

compute center of rotation

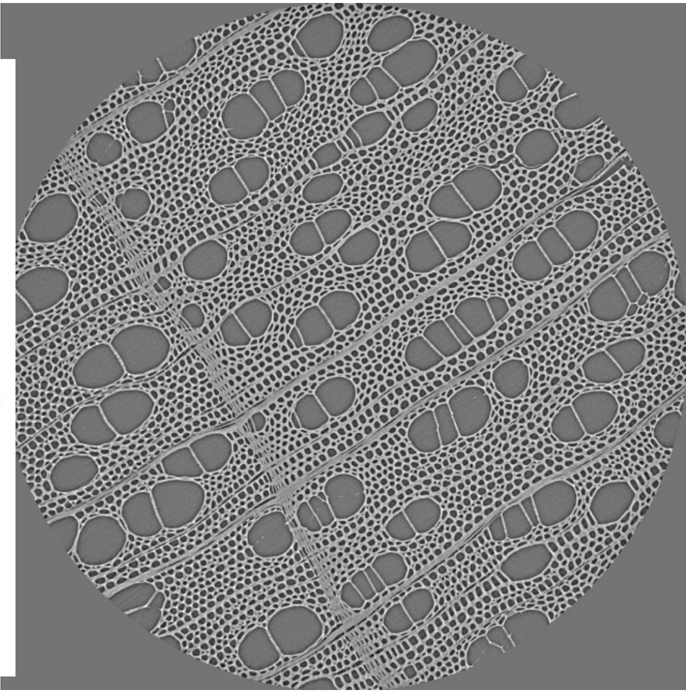
browse raw data, reconstructed slices, reduced darks / flats

compute reduced darks / flats

browse reconstructed volume

reconstruct a single slice

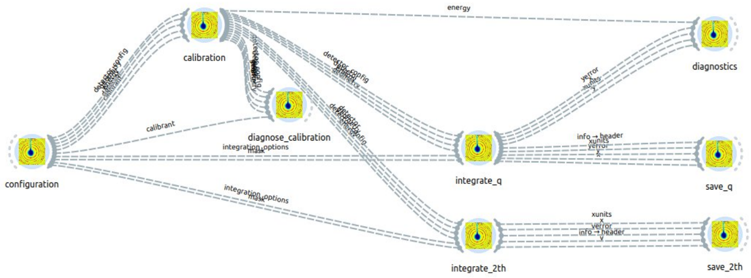
reconstruct a full volume



Inputs
Bliss scans
(darks, flats, projections + metadata)

Outputs
Reconstructed volumes

HOW WORKFLOWS CAN HELP

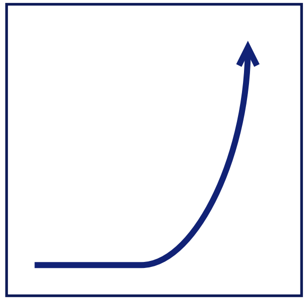


Reproducibility

Results can be regenerated by running the recipe

Abstracting complexity

Tasks can be rearranged or reused in other workflows without deep knowledge of the task content



Findable
Accessible
Interoperable
Re-Usable

Traceability

Workflows describe the history of a piece of data

More efficient beam time

FAIR data

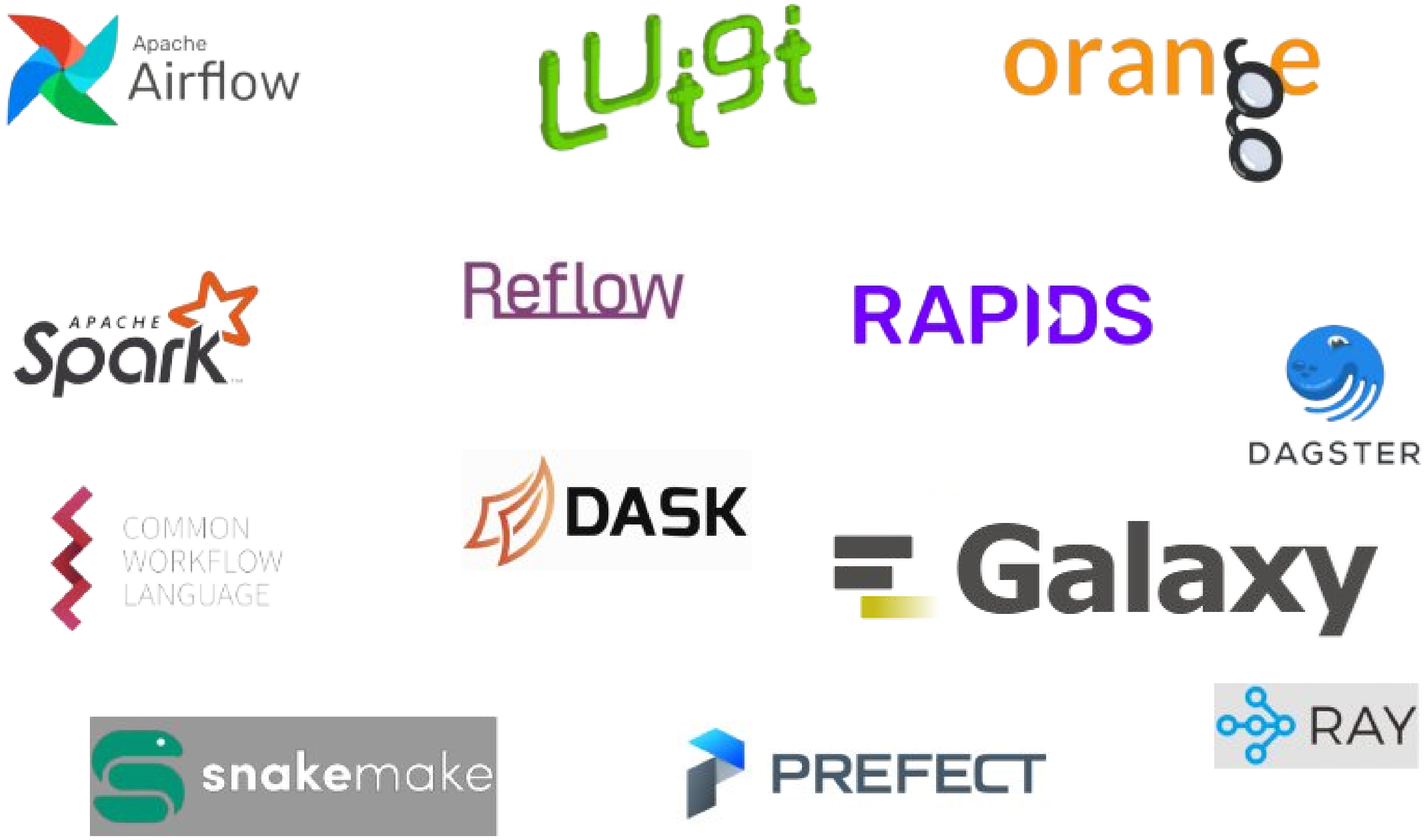
Reuseability

Process many datasets with the same recipe

Data are more valuable !

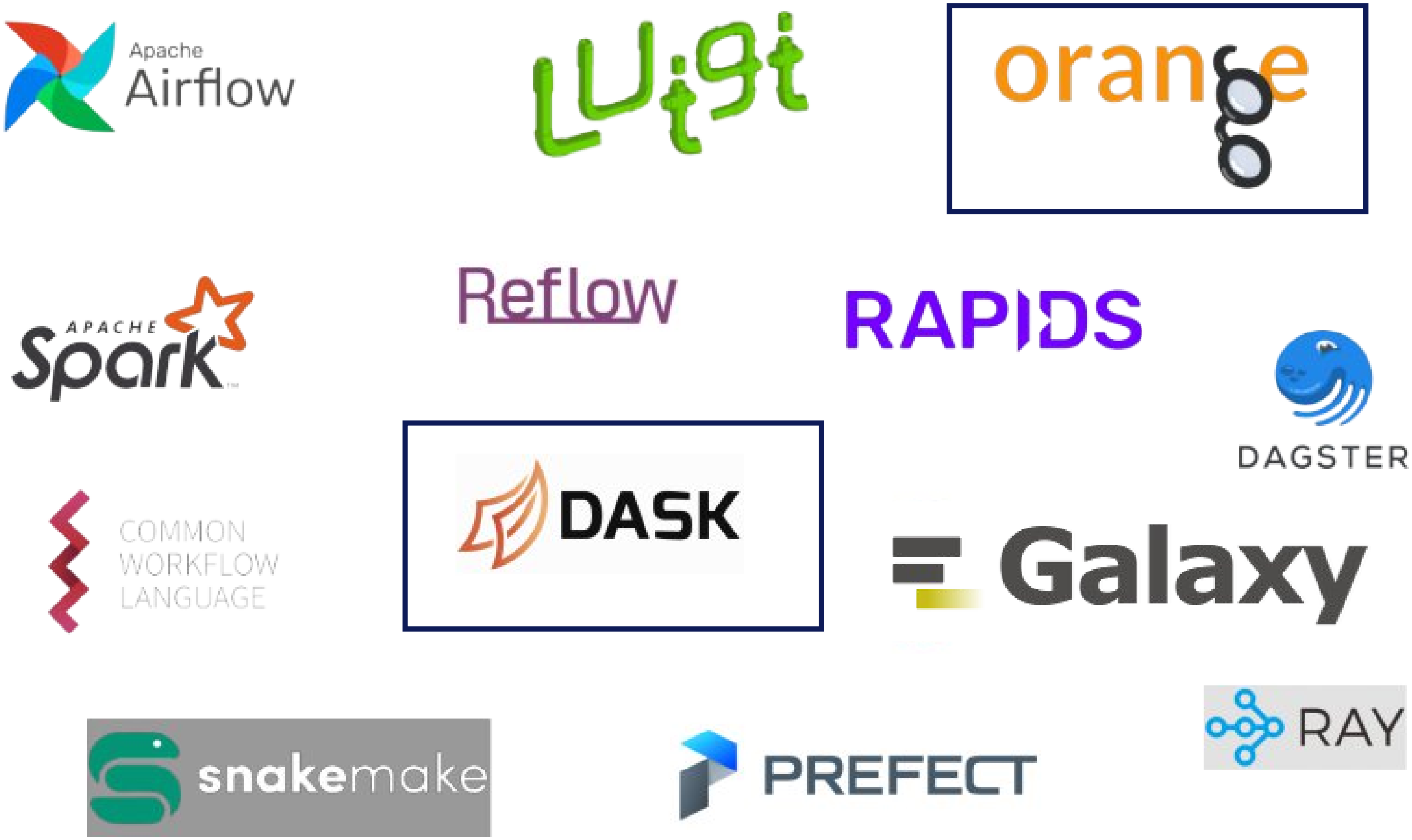
WORKFLOW SYSTEMS ARE (TOO) POPULAR

There are **hundreds** (yes, hundreds) of workflow systems !



...

SOME ARE ALREADY USED AT ESRF



+ Custom solutions

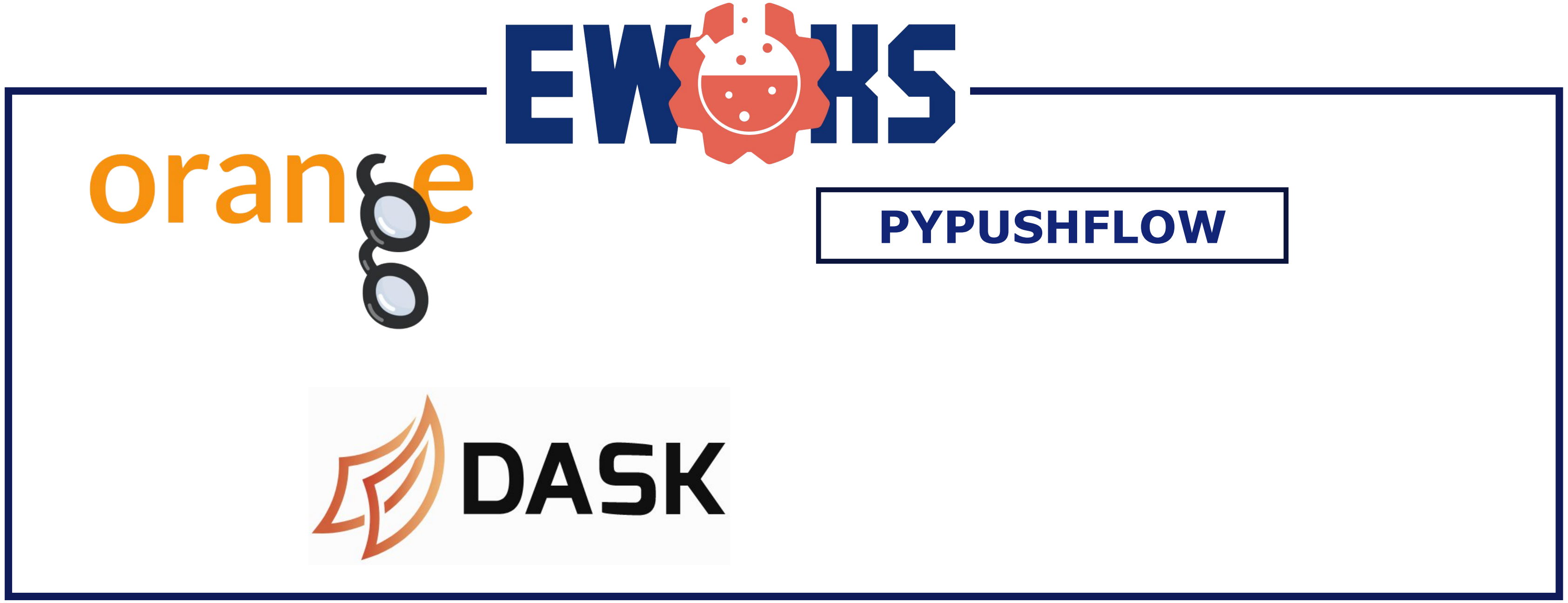
PYPUSHFLOW

How to provide **consistent data processing**

with all these **existing workflow systems** ?

THE EXTENSIBLE WORKFLOW SYSTEM EWOKS

We developed a “meta”-workflow system

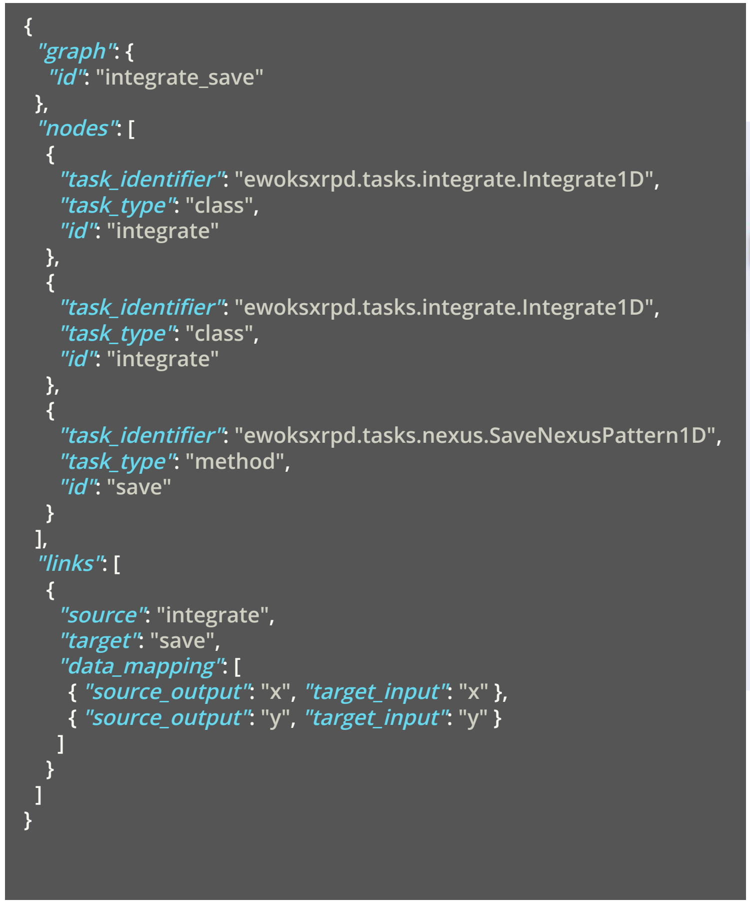


It can wrap any existing workflow system

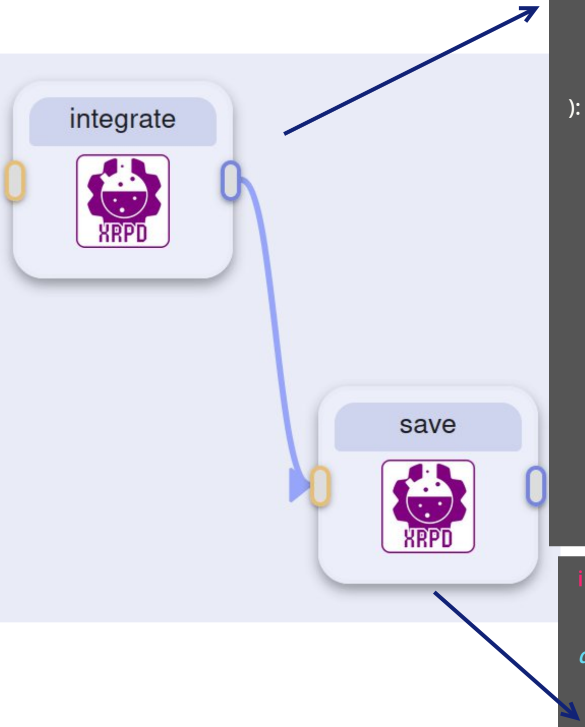
WHAT IS EWOKS

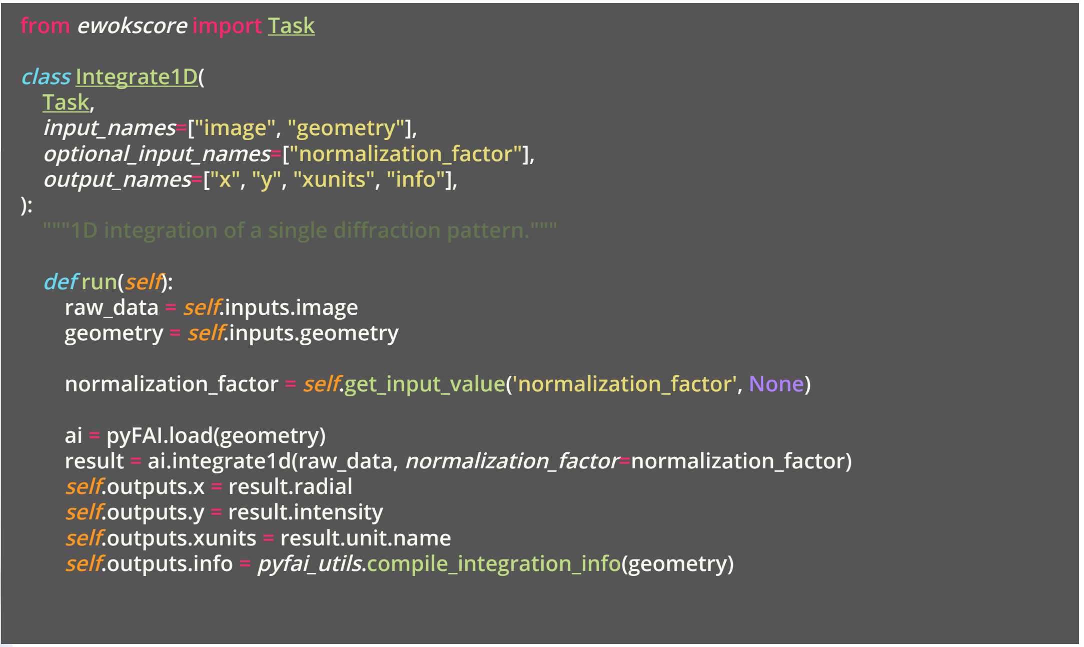
A common way of defining workflows across workflow systems

A common way of creating tasks across workflow systems

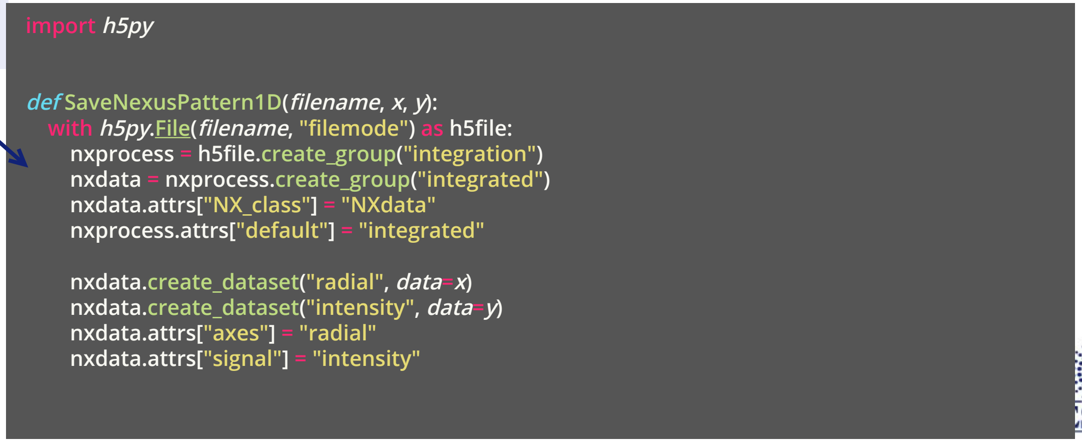


```
{
  "graph": {
    "id": "integrate_save"
  },
  "nodes": [
    {
      "task_identifier": "ewoksxrpd.tasks.integrate.Integrate1D",
      "task_type": "class",
      "id": "integrate"
    },
    {
      "task_identifier": "ewoksxrpd.tasks.integrate.Integrate1D",
      "task_type": "class",
      "id": "integrate"
    },
    {
      "task_identifier": "ewoksxrpd.tasks.nexus.SaveNexusPattern1D",
      "task_type": "method",
      "id": "save"
    }
  ],
  "links": [
    {
      "source": "integrate",
      "target": "save",
      "data_mapping": [
        { "source_output": "x", "target_input": "x" },
        { "source_output": "y", "target_input": "y" }
      ]
    }
  ]
}
```





```python
from ewokscore import Task

class Integrate1D(
    Task,
    input_names=["image", "geometry"],
    optional_input_names=["normalization_factor"],
    output_names=["x", "y", "xunits", "info"],
):
    """1D integration of a single diffraction pattern."""

    def run(self):
        raw_data = self.inputs.image
        geometry = self.inputs.geometry

        normalization_factor = self.get_input_value('normalization_factor', None)

        ai = pyFAI.load(geometry)
        result = ai.integrate1d(raw_data, normalization_factor=normalization_factor)
        self.outputs.x = result.radial
        self.outputs.y = result.intensity
        self.outputs.xunits = result.unit.name
        self.outputs.info = pyfai_utils.compile_integration_info(geometry)
```

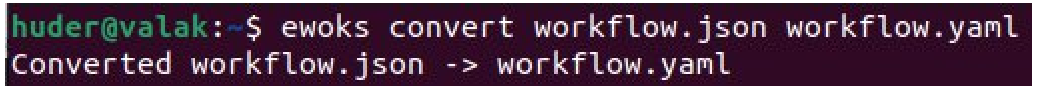


```python
import h5py


def SaveNexusPattern1D(filename, x, y):
    with h5py.File(filename, "filemode") as h5file:
        nxprocess = h5file.create_group("integration")
        nxdata = nxprocess.create_group("integrated")
        nxdata.attrs["NX_class"] = "NXdata"
        nxprocess.attrs["default"] = "integrated"

        nxdata.create_dataset("radial", data=x)
        nxdata.create_dataset("intensity", data=y)
        nxdata.attrs["axes"] = "radial"
        nxdata.attrs["signal"] = "intensity"
```

EWOKS PHILOSOPHY

Ewoks is

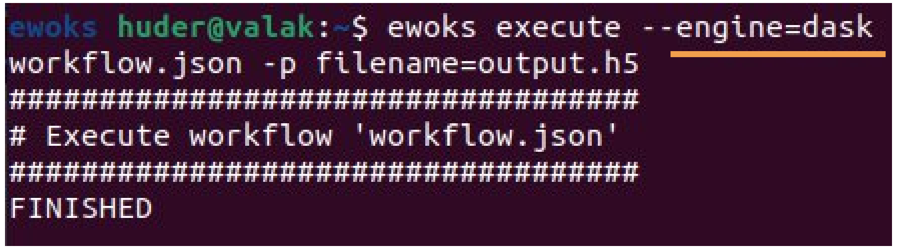
- A common way of defining workflows across workflow systems
- A common way of creating tasks across workflow systems
- A set of tools that are independent of workflow systems
  - Workflow serialization (JSON, YAML)
  - Logging of execution events
  - ...



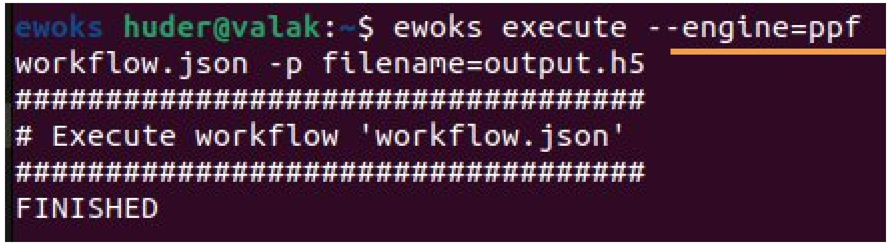
```
huder@valak:~$ ewoks convert workflow.json workflow.yaml
Converted workflow.json -> workflow.yaml
```

Ewoks is NOT

- Another workflow system
- Another workflow execution engine



```
ewoks huder@valak:~$ ewoks execute --engine=dask
workflow.json -p filename=output.h5
###################################
# Execute workflow 'workflow.json'
###################################
FINISHED
```

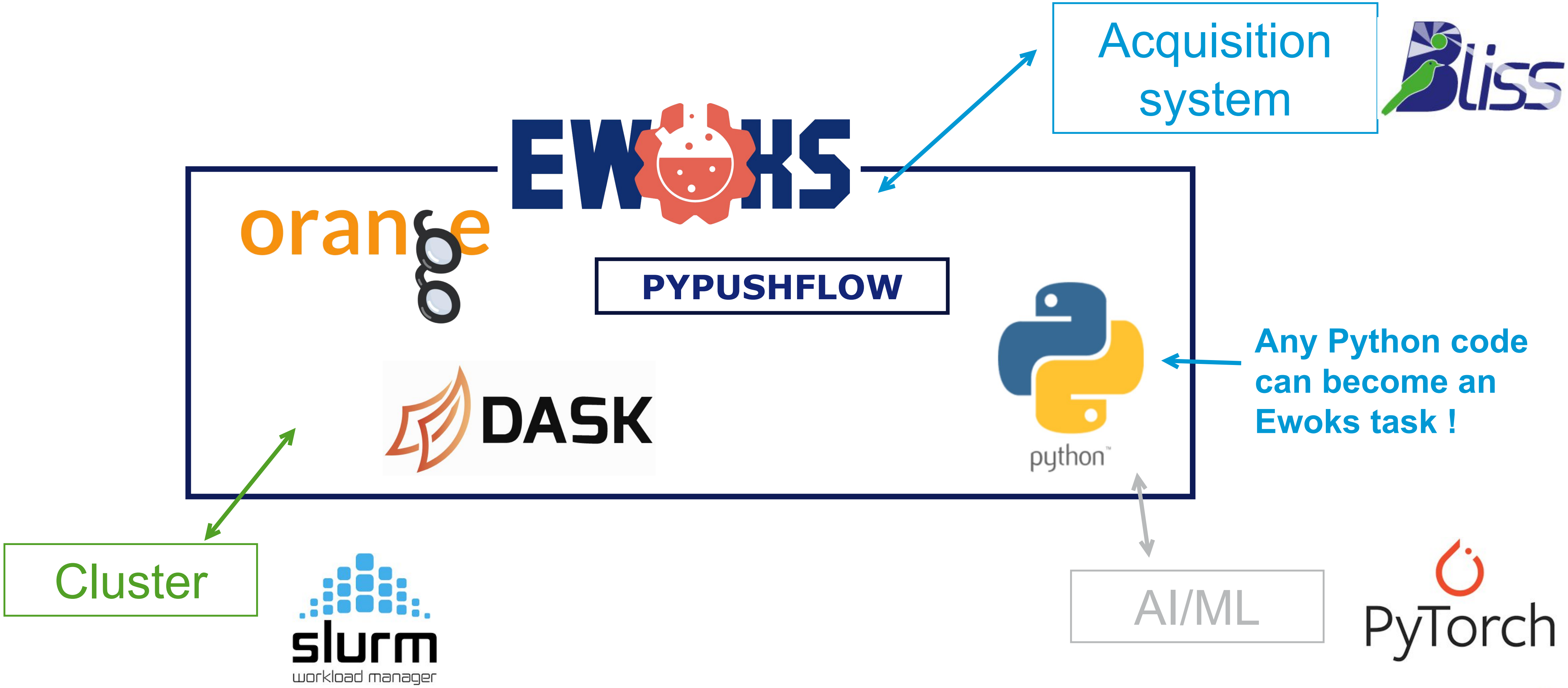


```
ewoks huder@valak:~$ ewoks execute --engine=ppf
workflow.json -p filename=output.h5
###################################
# Execute workflow 'workflow.json'
###################################
FINISHED
```

Ewoks decouples the workflow definition and task implementation from the workflow system used for the execution

EWOKS PHILOSOPHY

Ewoks can interact with established tools



AUTOMATED DATA PROCESSING

Acquisition control...

...triggers ewoks workflows...

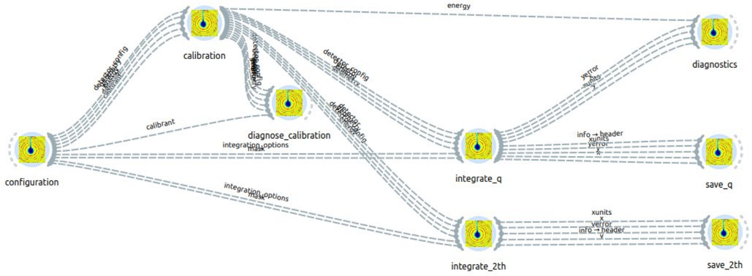
- on local machines
- on SLURM nodes

...producing processed data for...

Visualization

Persistence for further analysis

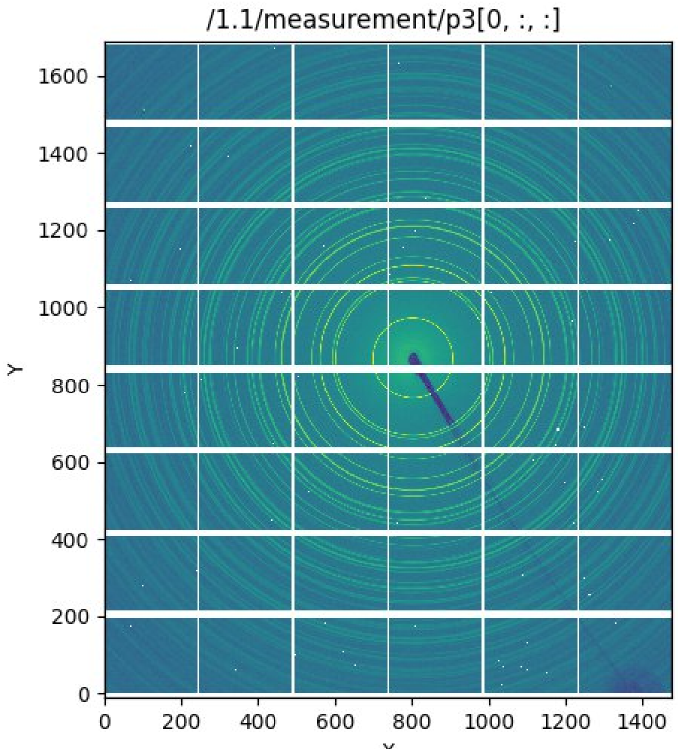
Uploading to Data portal
FAIR data

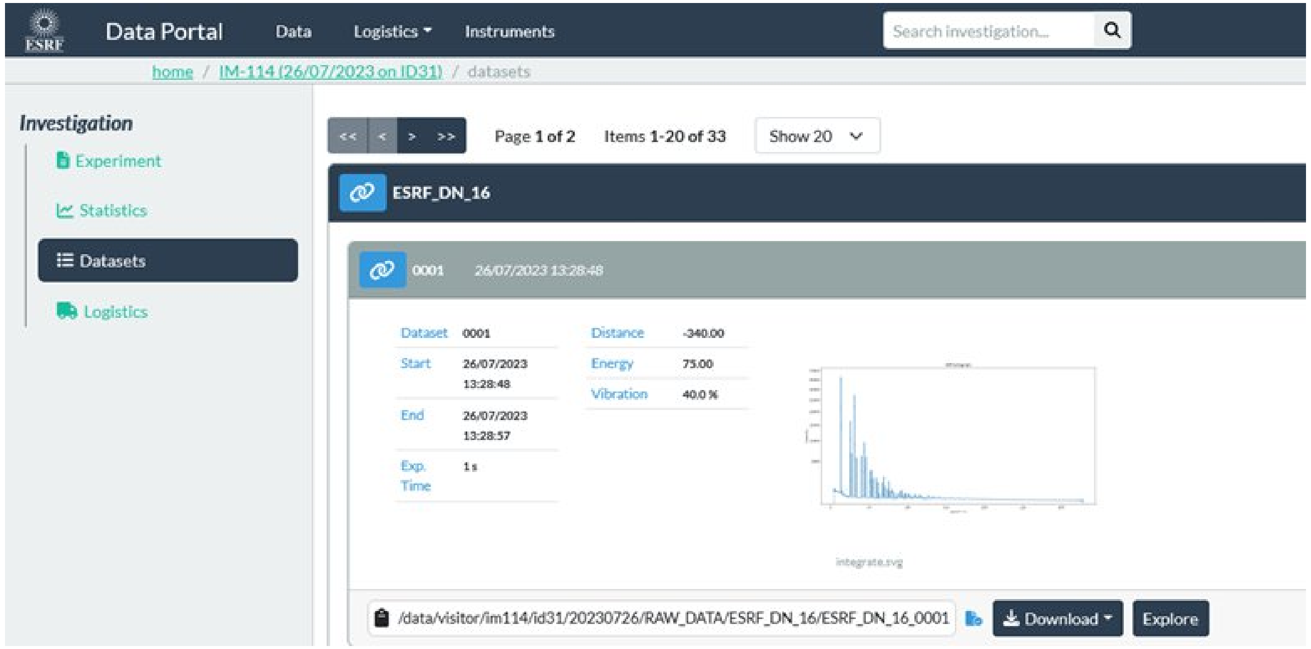


Datasets are processed as soon as they are acquired !

MAKING PROCESSED DATA AVAILABLE (DATA PORTAL)

Simple example: **Integrated patterns of X-ray powder diffraction images**





home / IM-114 (26/07/2023 on ID31) / datasets

Page 1 of 2 Items 1-20 of 33

ESRF_DN_16

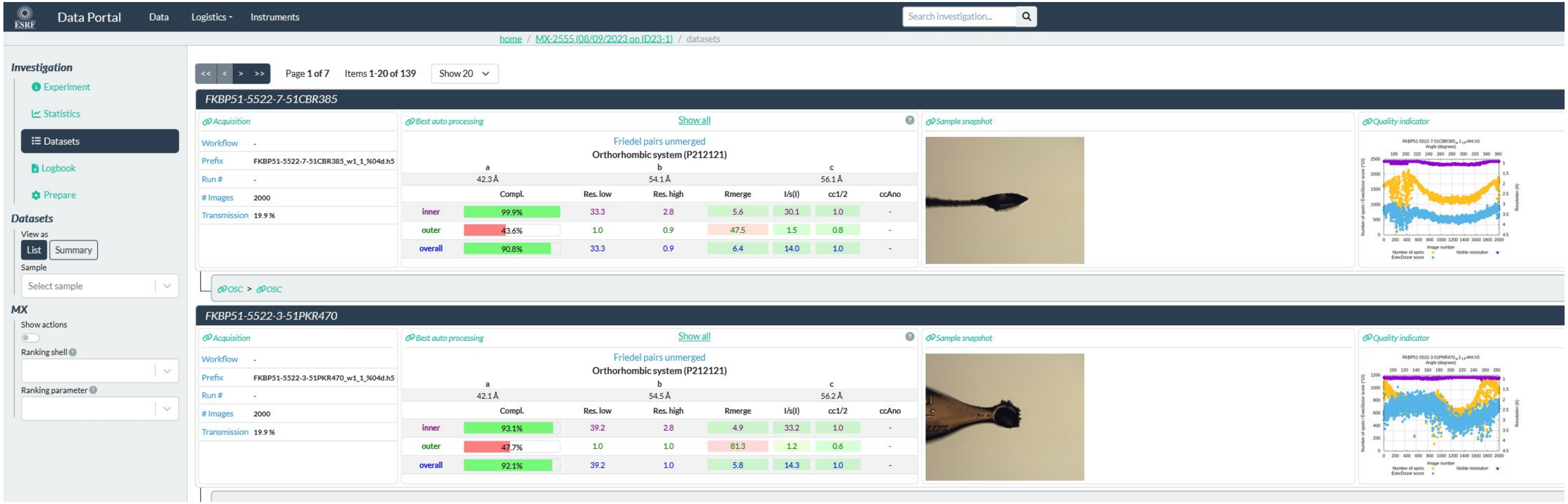
0001 26/07/2023 13:28:48

Dataset	0001	Distance	-340.00
Start	26/07/2023 13:28:48	Energy	75.00
End	26/07/2023 13:28:57	Vibration	40.0 %
Exp. Time	1 s		

/data/visitor/im114/id31/20230726/RAW_DATA/ESRF_DN_16/ESRF_DN_16_0001

MAKING PROCESSED DATA AVAILABLE (DATA PORTAL)

A bit more involved : **Diffraction of macromolecular crystals (MX) with diagnostics**



FKBP51-5522-7-51CBR385

Acquisition

Workflow	-
Prefix	FKBP51-5522-7-51CBR385_w1_1_%04d.h5
Run #	-
# Images	2000
Transmission	19.9 %

Best auto processing — Show all

Friedel pairs unmerged

Orthorhombic system (P212121)

a	b	c
42.3 Å	54.1 Å	56.1 Å

	Compl.	Res. low	Res. high	Rmerge	I/s(I)	cc1/2	ccAno
inner	99.9%	33.3	2.8	5.6	30.1	1.0	-
outer	43.6%	1.0	0.9	47.5	1.5	0.8	-
overall	90.8%	33.3	0.9	6.4	14.0	1.0	-

OSC > OSC

FKBP51-5522-3-51PKR470

Acquisition

Workflow	-
Prefix	FKBP51-5522-3-51PKR470_w1_1_%04d.h5
Run #	-
# Images	2000
Transmission	19.9 %

Best auto processing — Show all

Friedel pairs unmerged

Orthorhombic system (P212121)

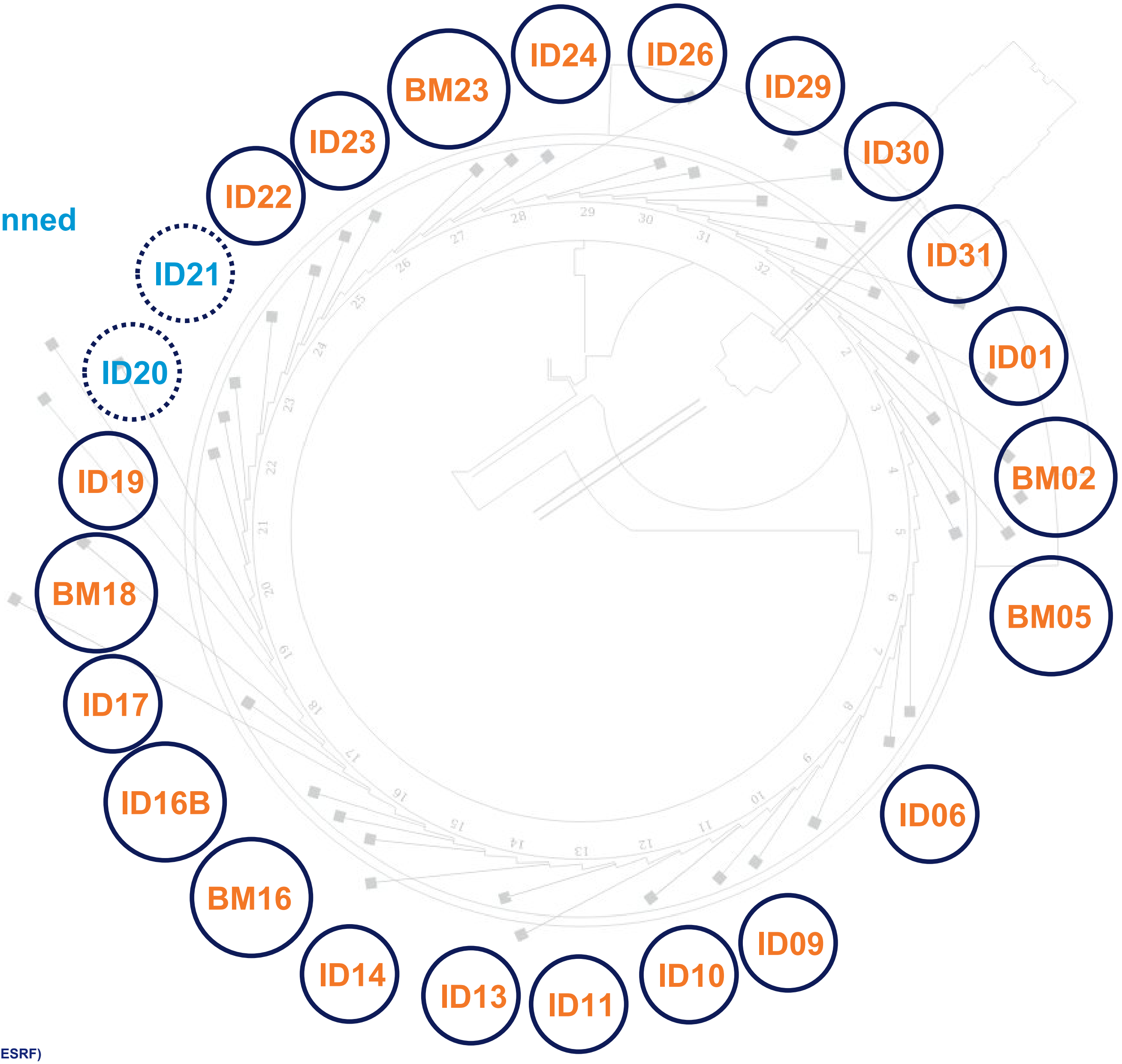
a	b	c
42.1 Å	54.5 Å	56.2 Å

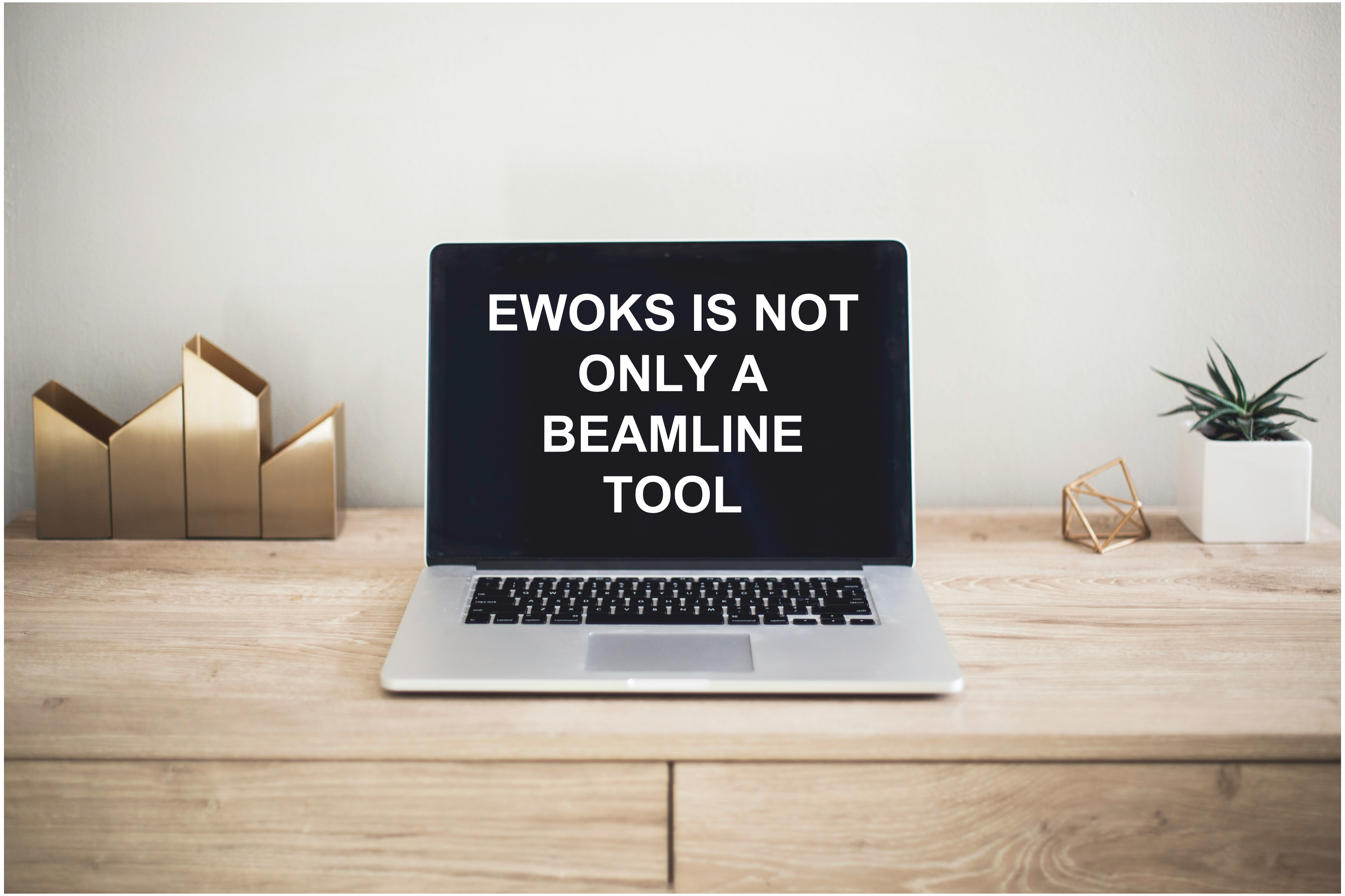
	Compl.	Res. low	Res. high	Rmerge	I/s(I)	cc1/2	ccAno
inner	93.1%	39.2	2.8	4.9	33.2	1.0	-
outer	47.7%	1.0	1.0	81.3	1.2	0.6	-
overall	92.1%	39.2	1.0	5.8	14.3	1.0	-

EWOKS AT ESRF

BM/ID In production

BM/ID In development/planned





Kari Shea / Unsplash

EWOKS IS NOT ONLY A BEAMLINE TOOL

Ewoks was designed to be used at beamlines but also by users on their local machine

The Ewoks ecosystem is made of Python packages

that can easily be installed locally by Python package managers:

```
pip install ewoks
```

Core packages to start using ewoks

```
pip install ewoksfluo
```

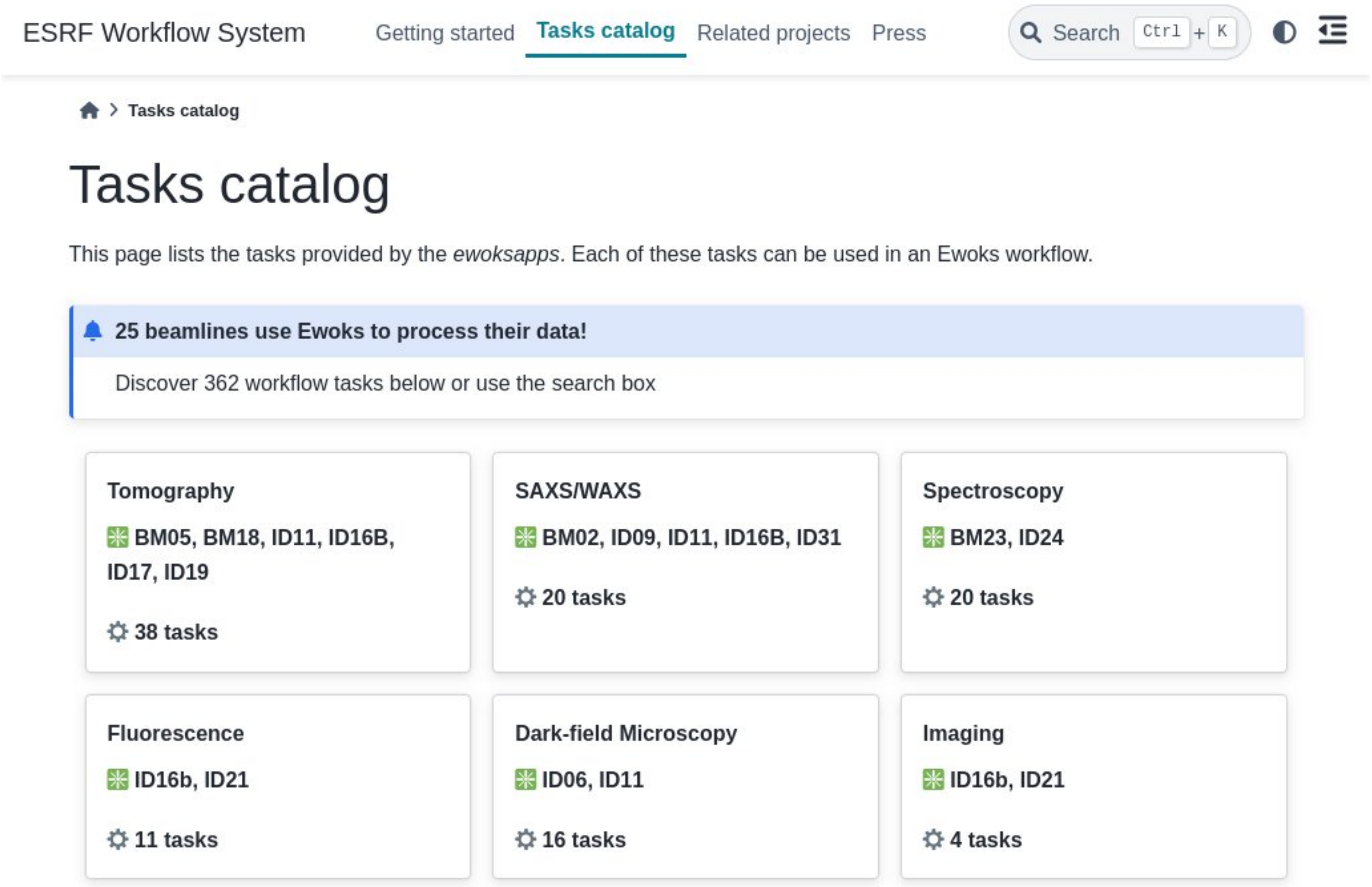
Install tasks for X-ray fluorescence data (e.g. peak fitting)

```
pip install ewoksxrpd
```

Install tasks for X-ray powder diffraction data (mostly integration with pyFAI)

And many more !

Ewoks tasks are searchable and findable on the Ewoks website by project or technique



Tasks catalog

This page lists the tasks provided by the *ewoksapps*. Each of these tasks can be used in an Ewoks workflow.

25 beamlines use Ewoks to process their data!

Discover 362 workflow tasks below or use the search box

Tomography
BM05, BM18, ID11, ID16B, ID17, ID19
38 tasks

SAXS/WAXS
BM02, ID09, ID11, ID16B, ID31
20 tasks

Spectroscopy
BM23, ID24
20 tasks

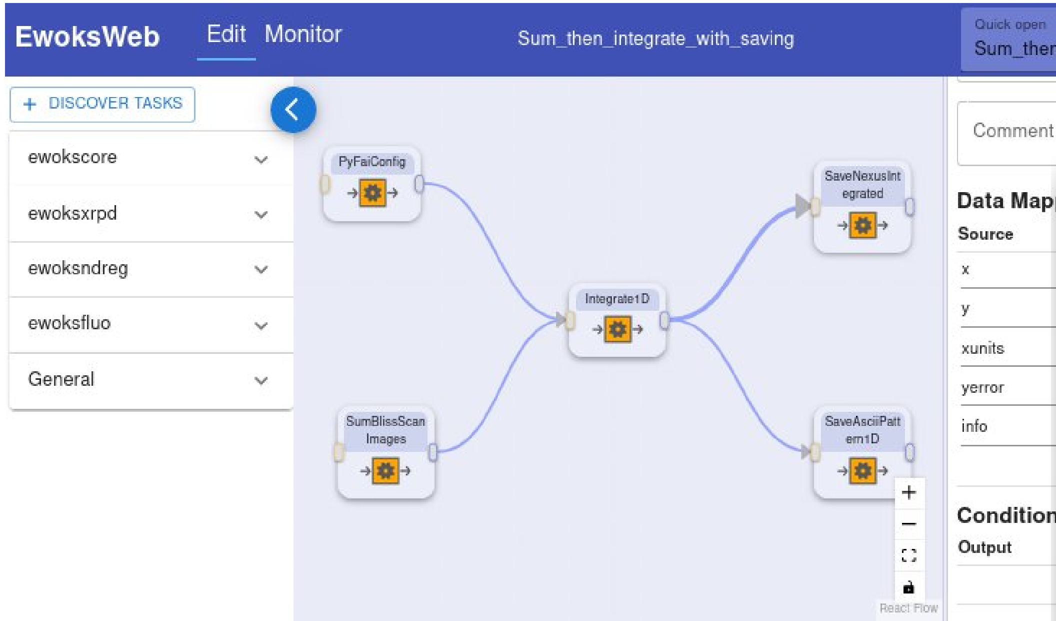
Fluorescence
ID16b, ID21
11 tasks

Dark-field Microscopy
ID06, ID11
16 tasks

Imaging
ID16b, ID21
4 tasks

https://ewoks.esrf.fr

EWOKS INTERFACES



EwoksWeb Edit Monitor

Sum_then_integrate_with_saving

Quick open

Sum_then_integrate ...

Comment

Data Mapp

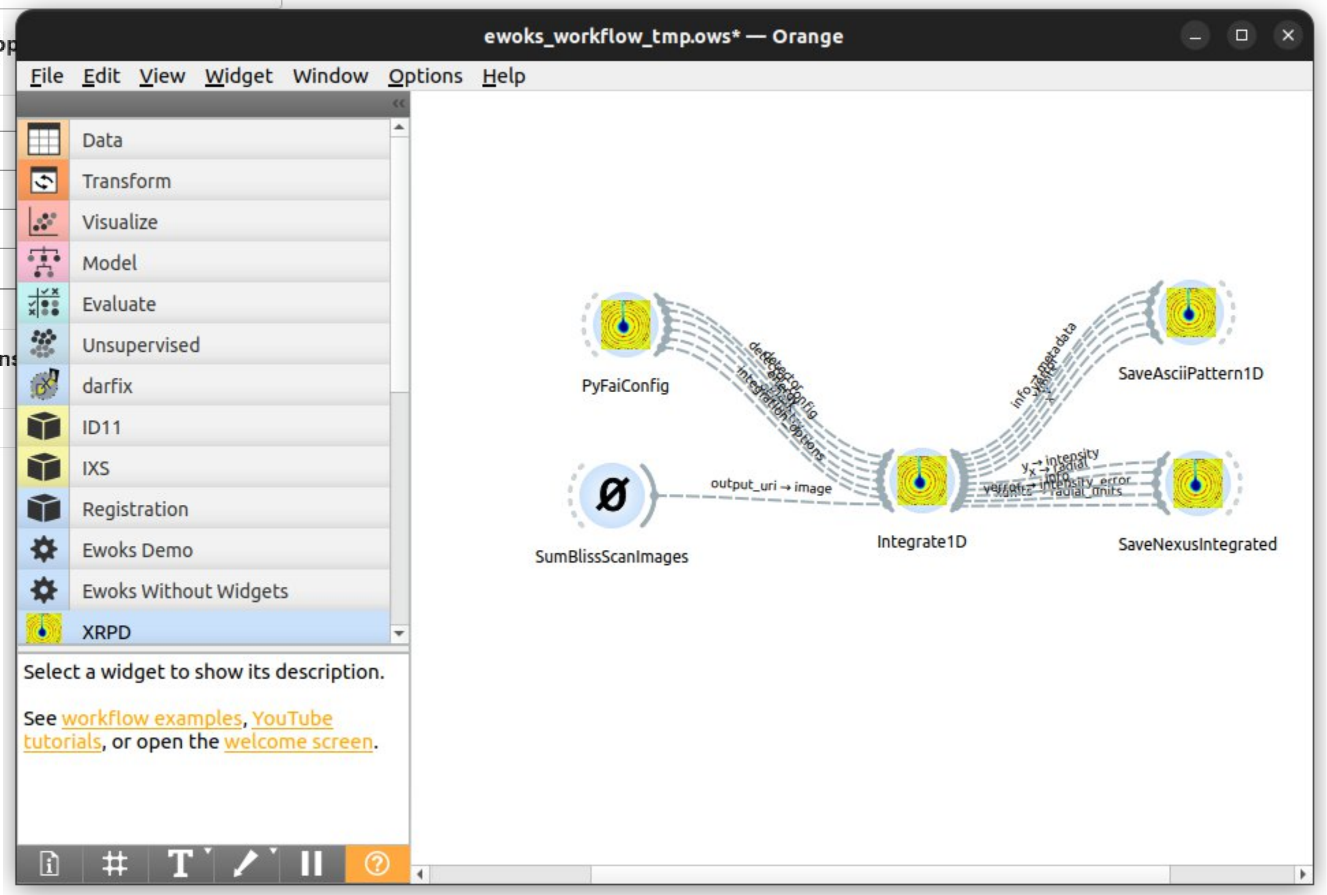
Source
x
y
xunits
yerror
info

Conditions

Output

EwoksWeb

Web GUI + REST server



ewoks_workflow_tmp.ows* — Orange

Select a widget to show its description.

See workflow examples, YouTube tutorials, or open the welcome screen.

CLI

Orange Canvas

Desktop GUI

MULTIPLE USE CASES, ONE WORKFLOW

USER

Writes or chooses existing tasks

Creates workflows

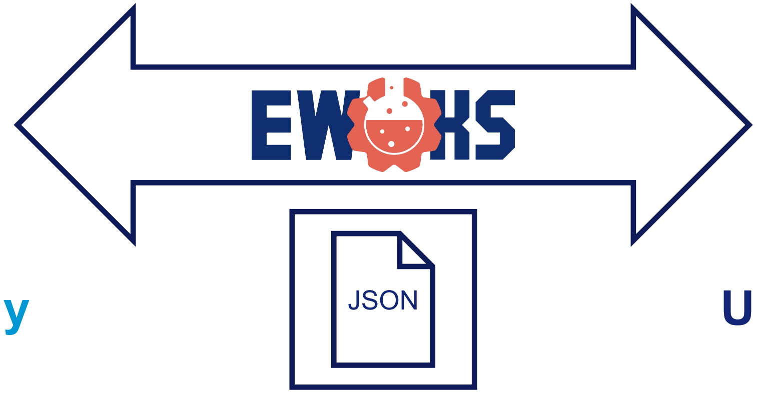
Processes the data locally

BEAMLINE

Uses existing tasks

Runs workflows automatically

Uploads the processed data to the data portal



```
{
  "graph": {"id": "integrate_save"},
  "nodes": [
    {"task_identifier":
"ewoksxrpd.tasks.integrate.Integrate1D", "task_type": "class",
"id": "integrate"},
    {"task_identifier":
"ewoksxrpd.tasks.nexus.SaveNexusPattern1D", "task_type":
"class", "id": "save"}
  ],
  "links": [{"source": "integrate", "target": "save"}]
}
```

CONCLUSION

EWOKS is a great asset to provide automatic data processing

EWOKS

🧑‍🔬 Can be used **locally** or **by beamlines and clusters**

Works across workflow systems and engines 🧑‍💻

Interacts with established tools
Acquisition systems and Python ecosystem

The success story continues

🚀

Used at 20+ ESRF beamlines
Many exciting developments to come
User support by a dedicated team

Questions ?

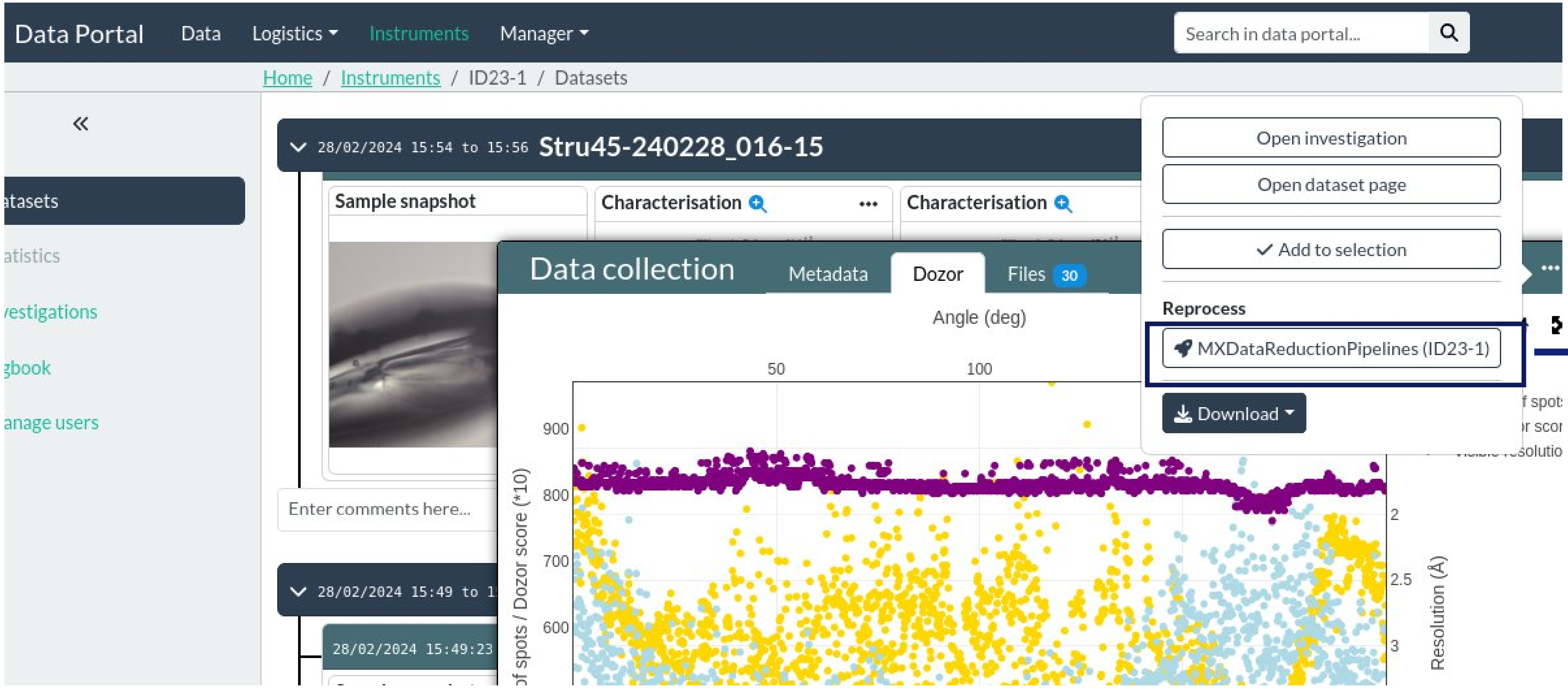
ESRF | The European Synchrotron

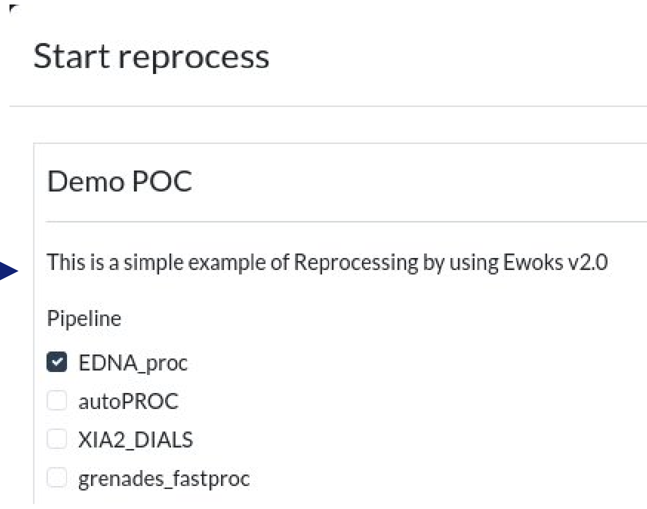
STREAMLINE has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 870313

https://streamline.esrf.fr/

A TEASER OF WHAT'S TO COME

Reprocessing in the data portal via a single click !





Start reprocess

Demo POC

This is a simple example of Reprocessing by using Ewoks v2.0

Pipeline

☑ EDNA_proc
□ autoPROC
□ XIA2_DIALS
□ grenades_fastproc

Only for MX (**for now...**)
Our goal is to do advanced processing for as many techniques as possible e.g. fluorescence mapping, tomography, 3DXRD, dark field microscopy, ...

SOME EWOKS EXAMPLES

Implement a workflow task

```
from ewokscore import Task

class SumTask(
    Task,
    input_names=["a"],
    optional_input_names=["b"],
    output_names=["result"]
):
    def run(self):
        result = self.inputs.a
        if self.inputs.b:
            result += self.inputs.b
        self.outputs.result = result
```

Execute a workflow

```
from ewoks import execute_graph

inputs = [{"id": "task1",
           "name": "a",
           "value": 10}]

# Optionally persist all task outputs
varinfo = {"root_uri": "/tmp/myresults"}

result = execute_graph(workflow,
                       varinfo=varinfo,
                       inputs=inputs)
print(result)
```

Define a workflow

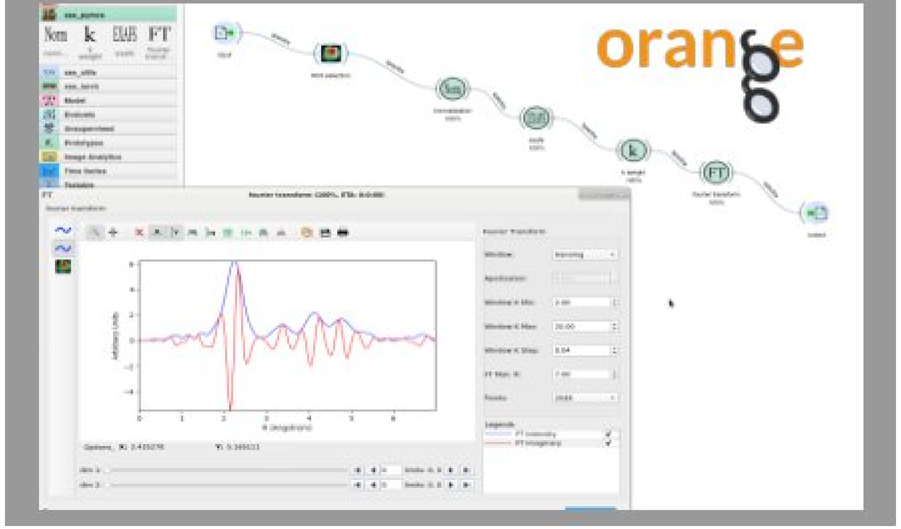
```
nodes = [
    {
        "id": "task1",
        "task_type": "class",
        "task_identifier": "mypackage.SumTask",
        "default_inputs": [{"name": "a",
                            "value": 1}],
    },
    {
        "id": "task2",
        "task_type": "class",
        "task_identifier": "mypackage.SumTask",
        "default_inputs": [{"name": "b",
                            "value": 1}],
    },
]
links = [
    {
        "source": "task1",
        "target": "task2",
        "data_mapping": [{"source_output": "result",
                          "target_input": "a"}],
    },
]
workflow = {"graph": {"id": "testworkflow"},
            "nodes": nodes,
            "links": links}
```

For more information
https://ewoks.readthedocs.io

ONLINE EXAFS PLOTTING

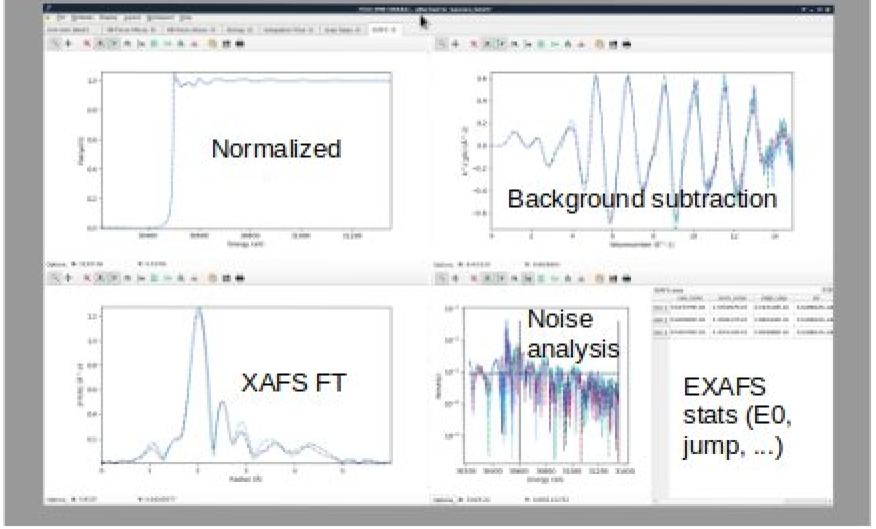
EWOKS (GUI)

Create workflow + choose parameters



FLINT

Plot results



Normalized

Background subtraction

XAFS FT

Noise analysis

EXAFS stats (E0, jump, ...)

Save workflow in a file

Every x seconds during scan: send workflow to worker and send result to Flint

EWOKS (worker)

Execute workflow

redis

Celery